Workgroup: Network Working Group
Internet-Draft: draft-dunglas-vulcain-00
Published: 3 April 2020
Intended Status: Informational
Expires: 5 October 2020
Authors: K. Dunglas
Les-Tilleuls.coop

The Vulcain Protocol

Abstract

This specification defines new HTTP headers (and query parameters) allowing a client to inform the server of the exact data it needs:

* Preload informs the server that relations of the main requested resource will be necessary. The server can then reduce the number of round-trips by sending the related resources ahead of time using HTTP/2 [RFC7540] Server Push. When using Server Push isn't possible (resources served by a different authority, server not supporting HTTP/2...), the server can hint the client to fetch those resources as early as possible by using the preload link relation [W3C.CR-preload-20171026] and the 103 status code [RFC8297].

* Fields informs the server of the list of fields of the retrieved resources that will be used. In order to improve performance and reduce bandwidth usage, the server can omit the fields not requested.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 October 2020.

Copyright Notice

Table of Contents

1. Terminology

2. Preload Header

  2.1. Using Preload Link Relations

3. Fields Header

4. Selectors

  4.1. Extended JSON Pointer

5. Query Parameters

6. Computing Links Server-Side

7. Security Considerations

8. IANA considerations

9. Normative References

10. Informative References

Author's Address

1. Terminology

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL**, when they appear in this document, are to be interpreted as described in [RFC2119].

2. Preload Header

Many formats including HTML [W3C.REC-html52-20171214], JSON-LD [W3C.REC-json-ld-20140116], Atom [RFC4287], XML [W3C.REC-xml-20081126], HAL and JSON:API allow the use of Web Linking [RFC5988] to represent references between resources.

The Preload HTTP header allows the client to ask the server to transmit resources linked to the main resource it will need as soon as possible. To do so, the Preload header MUST contain a selector [#selectors] referencing links to resources that SHOULD be preloaded.

The server MUST recursively follow links referenced by the selector. When a selector traverses several resources, all the traversed resources SHOULD be sent to the client. If several links referencing the same resource are selected, this resource MUST be sent at most once.

The server MAY limit the number resources that it sends in response to one request.

Multiple selectors can be sent by passing multiple Preload HTTP headers.

Considering the following resources:

/books

```
{
    "member": [
        "/books/1",
        "/books/2"
    ]
}
```

/books/1

```
{
    "title": "1984",
    "author": "/authors/1"
}
```

/books/2

```
{
    "title": "Homage to Catalonia",
    "author": "/authors/1"
}
```

```
   /authors/1

{
    "givenName": "George",
    "familyName": "Orwell"
}
```

```
   The Preload HTTP header can be used to ask the server to immediately
   push resources related to the requested one:

GET /books/ HTTP/2
Preload: /member/*/author
```

```
   In addition to /books, the server SHOULD use HTTP/2 Server Push to
   push the /books/1, /books/2 and /authors/1 resources. While it is
   referenced twice, /authors/1 MUST be pushed only once.

   Server Push requests generated by the server for related resources
   MUST include the remaining selector in a Preload HTTP header. When
   requesting a pushed relation, the client MUST compute the remaining
   selector and pass it in the Preload header.

   Example:

   Explicit Request:

GET /books/ HTTP/2
Preload: /member/*/author
```

```
   Request to a relation generated by the server (for the push) and the
   client:

GET /books/1 HTTP/2
Preload: /author
```

2.1. Using Preload Link Relations

If it's not possible or beneficial to use HTTP/2 Server Push (reference to a resource not served by the same authority, client or server not supporting HTTP/2, client having disabled Server Push...), preload link relations [W3C.CR-preload-20171026] SHOULD be used as a fallback.

The server MUST NOT add preload link relations if the related resources are pushed using HTTP/2 Server Push.

3. Fields Header

The Fields HTTP header allows the client to ask the server to return only the specified fields of the requested resource, and of the preloaded related resources.

The Fields HTTP header MUST contain a selector (see #Selector). The server SHOULD return only the fields matching this selector.

Multiple Fields HTTP headers can be passed. All fields matching at least one of these headers MUST be returned. Other fields of the resource MAY be omitted.

Considering the following resources:

/books/1

```
{
    "title": "1984",
    "genre": "novel",
    "author": "/authors/1"
}
```

/authors/1

```
{
    "givenName": "George",
    "familyName": "Orwell"
}
```

And the following HTTP request:

```
GET /books/1 HTTP/2
Preload: /author
Fields: /author/familyName
Fields: /genre
```

The server must return a response containing the following JSON document:

```
{
    "genre": "novel",
    "author": "/authors/1"
}
```

And push the following filtered /authors/1 resource:

```
{
    "familyName": "Orwell"
}
```

Server Push requests generated by the server for related resources MUST include the remaining selector in a Fields HTTP header. When requesting a pushed relation, the client MUST compute the remaining selector and pass it in the Fields header.

Example:

Explicit Request:

```
GET /books/ HTTP/2
Fields: /member/*/author
```

Request to a relation generated by the server (for the push) and the client:

```
GET /books/1 HTTP/2
Fields: /author
```

4. Selectors

Selectors used as value of the Preload and Fields HTTP headers depend on the Content-Type of the requested resource. This specification defines default selector formats for common content-types, and a mechanism to use other selector formats.

The client SHOULD use the Accept HTTP header to request the resource in a format compatible with selectors used in Preload and Fields HTTP headers.

The client can use the Prefer HTTP header [RFC7240] with the selector preference to ask the server to use a specific selector format:

```
GET /books/1 HTTP/2
Accept: text/xml
Prefer: selector=css
Fields: brand > name
```

If no explicit preferences have been passed, the server MUST assume that the selector format is the default corresponding to the format of the resource.

The following table defines the default selector format for common formats:

Format	Selector format	Identifier
JSON	Extended JSON Pointer Section 4.1	json-pointer
XML	XPath [W3C.REC-xpath-19991116]	xpath
HTML	CSS selectors [W3C.REC-selectors-3-20181106]	css

Table 1

The client and the server can negotiate the use of other selector formats using the Prefer HTTP header.

4.1. Extended JSON Pointer

For JSON documents, the default selector format is JSON Pointer [RFC6901]. However, JSON Pointer doesn't provide a mechanism to select entire collections.

This specification defines an extension to the JSON Pointer format allowing to select every element of a collection, the * character.

Considering the following JSON document:

```
{
    "books": [
        {
            "title": "1984",
            "author": "George Orwell"
        },
        {
            "title": "The Handmaid's Tale",
            "author": "Margaret Atwood"
        }
    ]
}
```

The /books/*/author JSON Pointer selects the author field of every objects in the books array.

The * character is escaped by encoding it as the ~2 character sequence.

By design, this selector is simple and limited. Simple selectors make it easier to limit the complexity of requests executed by the server.

5. Query Parameters

Another option available to clients is to utilize Request URI query-string parameters to pass preload and fields selectors. The preload and query parameters MAY be used to pass selectors corresponding respectively to the Preload and Fields HTTP headers. To pass multiple selectors, parameters can be passed multiple times.

Example: /books/1?fields=/title&fields=/author&preload=/author

When using query parameters, the server MUST pass the remaining part of the selector as parameter of the generated link.

Example:

```
GET /books/?preload=/member/*/author HTTP/2

{
    "member": {
        "/books/1?preload=/author",
        "/books/1?preload=/author"
    }
}
```

As altering the URI can have undesirable effects, using HTTP headers SHOULD be preferred. Support for query parameters is OPTIONAL. A server supporting query parameters MUST also support the corresponding HTTP headers.

6. Computing Links Server-Side

While using hypermedia capabilities of the HTTP protocol through Web Linking SHOULD always be preferred, sometimes links between resources are known by the server but are not provided in the HTTP response.

In such cases, the server can compute the link server-side in order to push the related resource. Such server-side computed links MAY be documented, for instance by providing an OpenAPI specification containing Link objects.

Considering the following resources and assuming that the server knows that the author field references the resources /authors/{id} resource:

/books/1

```
{
    "title": "1984",
    "author": 1
}
```

/authors/1

```
{
    "givenName": "George",
    "familyName": "Orwell"
}
```

In response to this request , both /books/1 and /authors/1 should be pushed:

```
GET /books/1 HTTP/2
Preload: /author
```

7. Security Considerations

Using the Preload header can lead to a large number of resources to be generated and pushed. The server SHOULD limit the maximum number of resources to push. The depth of the selector SHOULD also be limited by the server.

8. IANA considerations

The Preloadand Fields header fields will be added to the "Permanent Message Header Field Names" registry defined in [RFC3864].

A selector registry could also be added.

9. Normative References

[RFC8297] Oku, K., "An HTTP Status Code for Indicating Hints", RFC 8297, DOI 10.17487/RFC8297, December 2017, <https://www.rfc-editor.org/info/rfc8297>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <https://www.rfc-editor.org/info/rfc5988>.

[W3C.REC-selectors-3-20181106] Ã elik, T., Etemad, E., Glazman, D., Hickson, I., Linss, P., and J. Williams, "Selectors Level 3", World Wide Web Consortium Recommendation REC-selectors-3-20181106, 6 November 2018, <https://www.w3.org/TR/2018/REC-selectors-3-20181106>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[W3C.CR-preload-20171026] Grigorik, I. and Y. Weiss, "Preload", World Wide Web Consortium CR CR-preload-20171026, 26 October 2017, <https://www.w3.org/TR/2017/CR-preload-20171026>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[W3C.REC-xpath-19991116] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, 16 November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC7240] Snell, J., "Prefer Header for HTTP", RFC 7240, DOI 10.17487/RFC7240, June 2014, <https://www.rfc-editor.org/info/rfc7240>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

10. Informative References

[W3C.REC-html52-20171214] Faulkner, S., Eicholz, A., Leithead, T., Danilo, A., and S. Moon, "HTML 5.2", World Wide Web Consortium Recommendation REC-html52-20171214, 14 December 2017, <https://www.w3.org/TR/2017/REC-html52-20171214>.

[W3C.REC-json-ld-20140116] Sporny, M., Kellogg, G., and M. Lanthaler, "JSON-LD 1.0", World Wide Web Consortium Recommendation REC-json-ld-20140116, 16 January 2014, <http://www.w3.org/TR/2014/REC-json-ld-20140116>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, 26 November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <https://www.rfc-editor.org/info/rfc4287>.

Author's Address

Kévin Dunglas
Les-Tilleuls.coop
2 rue Hegel
59000 Lille
France

Email: kevin@les-tilleuls.coop